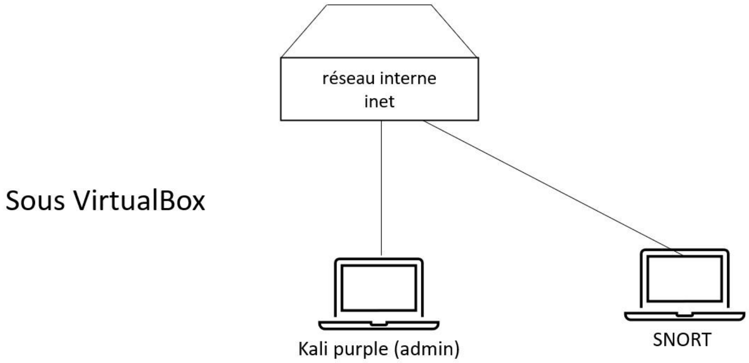
SECS1028 - Laboratoire 10 - Snort IDS/IPS

laboratoire not´e sur 11 points - 10% de la note finale

Objectif du laboratoire : Activer la protection IDS/IPS Snort sur un r´eseau interne VirtualBox.

Pour ce laboratoire, utilisez une VM Kali et une mouvelle VM Alpine Linux nomm´ee snort sur le r´eseau interne :



1 Installation de Snort (2 points)

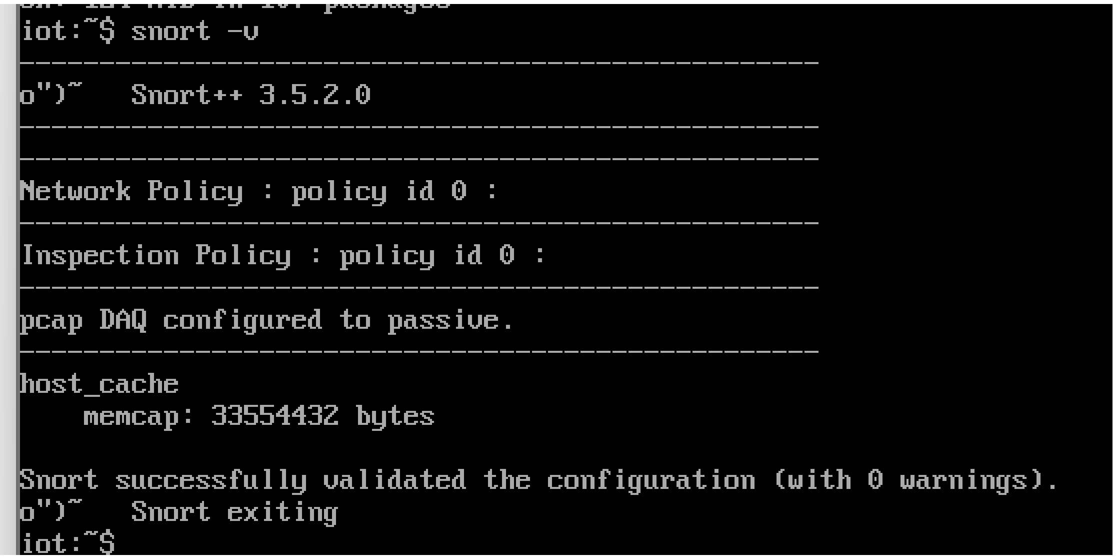
1) Qu'est ce que Snort ? (1 point)

   Snort est un système open source de détection et de prévention des intrusions (IDS/IPS) conçu pour analyser en temps réel le trafic réseau et détecter les activités potentiellement malveillantes

2) Cr´eez une nouvelle VM Alpine Linux avec un disque DVI de 1Go connect´ee en NAT pendant l'installation. Puis Installez Snort sur cette VM :

```
sudo apk update sudo
apk add snort
```

Capture ´ecran de la version de snort (snort -v) (1 point) :



```
iot:~$ snort -v
--------------------------------------------------
o")~   Snort++ 3.5.2.0
--------------------------------------------------
--------------------------------------------------
Network Policy : policy id 0 :
--------------------------------------------------
Inspection Policy : policy id 0 :
--------------------------------------------------
pcap DAQ configured to passive.
--------------------------------------------------
host_cache
    memcap: 33554432 bytes

Snort successfully validated the configuration (with 0 warnings).
o")~   Snort exiting
iot:~$
```

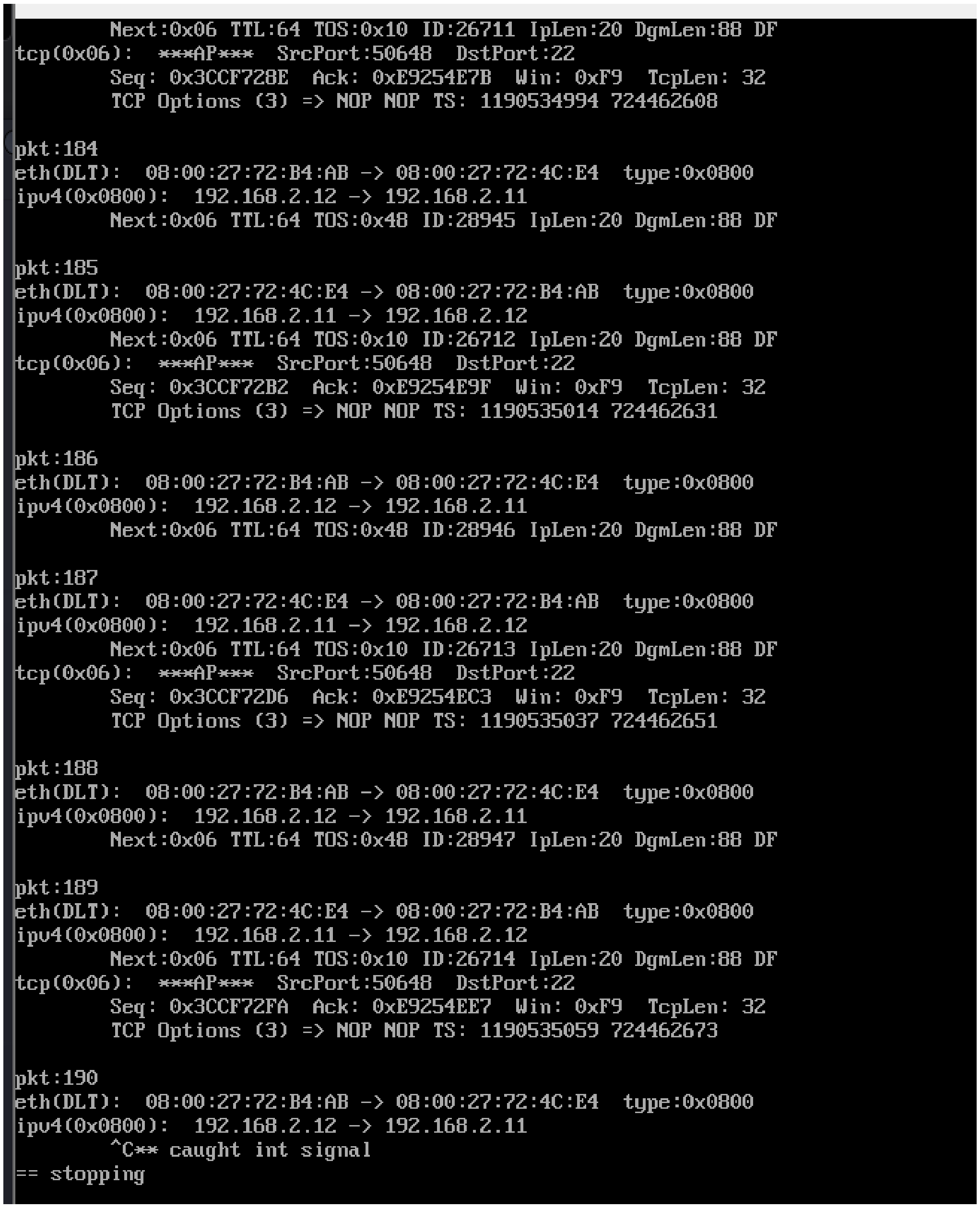
Puis installez tshark et netcat : sudo apk add tshark netcat-openbsd

2 Snort en mode IDS (9 points)

Connectez `a pr´esent la VM snort sur un r´eseau interne de VirtualBox. Pour ce laboratoire, vous pouvez utiliser un shell directement sur la VM snort ou un shell sur Kali connect´ee en ssh sur la VM snort.

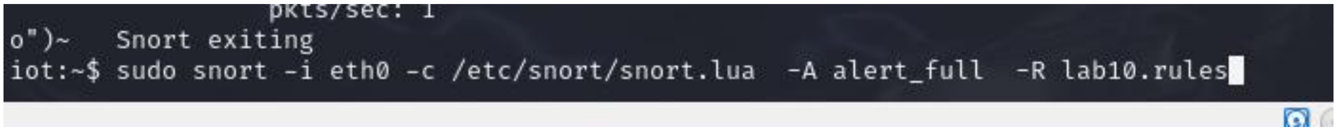
1) La commande Snort pour afficher sur le terminal les paquets r´eseaux est sudo snort -i <interface r´eseau> -L dump capture

´ecran : (1 point)



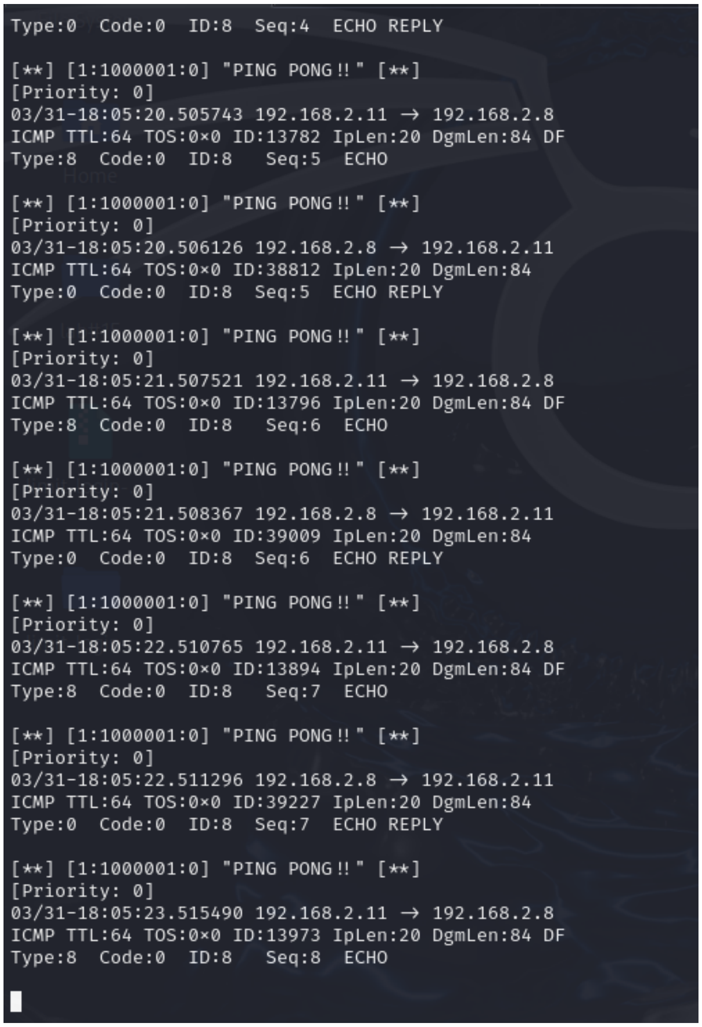
```
        Next:0x06 TTL:64 TOS:0x10 ID:26711 IpLen:20 DgmLen:88 DF
tcp(0x06):  ***AP***  SrcPort:50648  DstPort:22
        Seq: 0x3CCF728E  Ack: 0xE9254E7B  Win: 0xF9  TcpLen: 32
        TCP Options (3) => NOP NOP TS: 1190534994 724462608

pkt:184
eth(DLT):  08:00:27:72:B4:AB -> 08:00:27:72:4C:E4  type:0x0800
ipv4(0x0800):  192.168.2.12 -> 192.168.2.11
        Next:0x06 TTL:64 TOS:0x48 ID:28945 IpLen:20 DgmLen:88 DF

pkt:185
eth(DLT):  08:00:27:72:4C:E4 -> 08:00:27:72:B4:AB  type:0x0800
ipv4(0x0800):  192.168.2.11 -> 192.168.2.12
        Next:0x06 TTL:64 TOS:0x10 ID:26712 IpLen:20 DgmLen:88 DF
tcp(0x06):  ***AP***  SrcPort:50648  DstPort:22
        Seq: 0x3CCF72B2  Ack: 0xE9254E9F  Win: 0xF9  TcpLen: 32
        TCP Options (3) => NOP NOP TS: 1190535014 724462631

pkt:186
eth(DLT):  08:00:27:72:B4:AB -> 08:00:27:72:4C:E4  type:0x0800
ipv4(0x0800):  192.168.2.12 -> 192.168.2.11
        Next:0x06 TTL:64 TOS:0x48 ID:28946 IpLen:20 DgmLen:88 DF

pkt:187
eth(DLT):  08:00:27:72:4C:E4 -> 08:00:27:72:B4:AB  type:0x0800
ipv4(0x0800):  192.168.2.11 -> 192.168.2.12
        Next:0x06 TTL:64 TOS:0x10 ID:26713 IpLen:20 DgmLen:88 DF
tcp(0x06):  ***AP***  SrcPort:50648  DstPort:22
        Seq: 0x3CCF72D6  Ack: 0xE9254EC3  Win: 0xF9  TcpLen: 32
        TCP Options (3) => NOP NOP TS: 1190535037 724462651

pkt:188
eth(DLT):  08:00:27:72:B4:AB -> 08:00:27:72:4C:E4  type:0x0800
ipv4(0x0800):  192.168.2.12 -> 192.168.2.11
        Next:0x06 TTL:64 TOS:0x48 ID:28947 IpLen:20 DgmLen:88 DF

pkt:189
eth(DLT):  08:00:27:72:4C:E4 -> 08:00:27:72:B4:AB  type:0x0800
ipv4(0x0800):  192.168.2.11 -> 192.168.2.12
        Next:0x06 TTL:64 TOS:0x10 ID:26714 IpLen:20 DgmLen:88 DF
tcp(0x06):  ***AP***  SrcPort:50648  DstPort:22
        Seq: 0x3CCF72FA  Ack: 0xE9254EE7  Win: 0xF9  TcpLen: 32
        TCP Options (3) => NOP NOP TS: 1190535059 724462673

pkt:190
eth(DLT):  08:00:27:72:B4:AB -> 08:00:27:72:4C:E4  type:0x0800
ipv4(0x0800):  192.168.2.12 -> 192.168.2.11
        ^C** caught int signal
== stopping
```

2) Cr´eez une r`egle snort qui alerte les paquets de type ICMP. Quelle est cette regle ? (1 point)



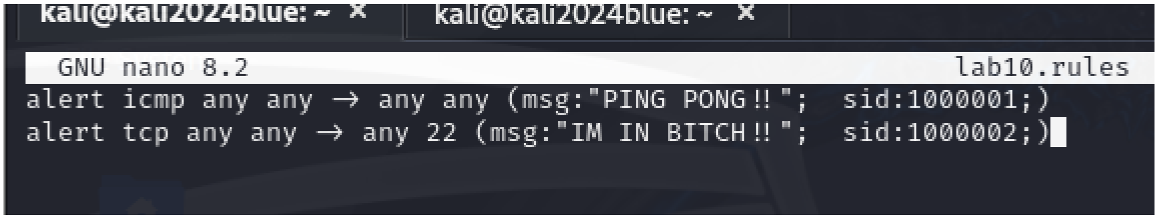
```
                 pkts/sec: 1
o")~   Snort exiting
iot:~$ sudo snort -i eth0 -c /etc/snort/snort.lua  -A alert_full  -R lab10.rules
```

3) Testez cette r`egle avec snort. Montrez qu'elle fonctionne en utilisant la commande ping. Capture ´ecran de l'alerte snort : (1 point)



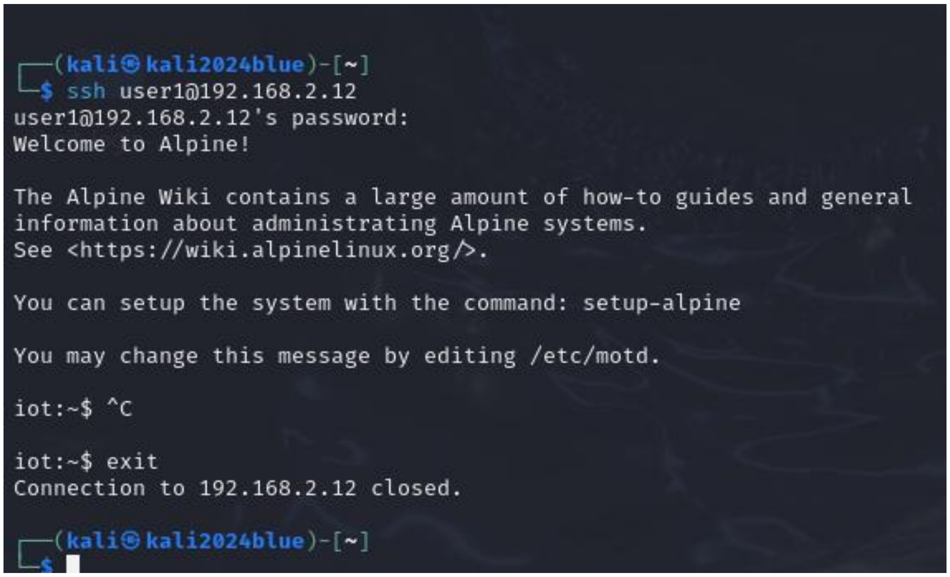
```
Type:0  Code:0  ID:8  Seq:4  ECHO REPLY

[**] [1:1000001:0] "PING PONG!!" [**]
[Priority: 0]
03/31-18:05:20.505743 192.168.2.11 -> 192.168.2.8
ICMP TTL:64 TOS:0x0 ID:13782 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:8   Seq:5  ECHO

[**] [1:1000001:0] "PING PONG!!" [**]
[Priority: 0]
03/31-18:05:20.506126 192.168.2.8 -> 192.168.2.11
ICMP TTL:64 TOS:0x0 ID:38812 IpLen:20 DgmLen:84
Type:0  Code:0  ID:8  Seq:5  ECHO REPLY

[**] [1:1000001:0] "PING PONG!!" [**]
[Priority: 0]
03/31-18:05:21.507521 192.168.2.11 -> 192.168.2.8
ICMP TTL:64 TOS:0x0 ID:13796 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:8   Seq:6  ECHO

[**] [1:1000001:0] "PING PONG!!" [**]
[Priority: 0]
03/31-18:05:21.508367 192.168.2.8 -> 192.168.2.11
ICMP TTL:64 TOS:0x0 ID:39009 IpLen:20 DgmLen:84
Type:0  Code:0  ID:8  Seq:6  ECHO REPLY

[**] [1:1000001:0] "PING PONG!!" [**]
[Priority: 0]
03/31-18:05:22.510765 192.168.2.11 -> 192.168.2.8
ICMP TTL:64 TOS:0x0 ID:13894 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:8   Seq:7  ECHO

[**] [1:1000001:0] "PING PONG!!" [**]
[Priority: 0]
03/31-18:05:22.511296 192.168.2.8 -> 192.168.2.11
ICMP TTL:64 TOS:0x0 ID:39227 IpLen:20 DgmLen:84
Type:0  Code:0  ID:8  Seq:7  ECHO REPLY

[**] [1:1000001:0] "PING PONG!!" [**]
[Priority: 0]
03/31-18:05:23.515490 192.168.2.11 -> 192.168.2.8
ICMP TTL:64 TOS:0x0 ID:13973 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:8   Seq:8  ECHO
```

4) Cr´eez une r`egle snort qui alerte les paquets entrants de type TCP sur le port 22 (ssh). Quelle est cette regle ? (1 point)

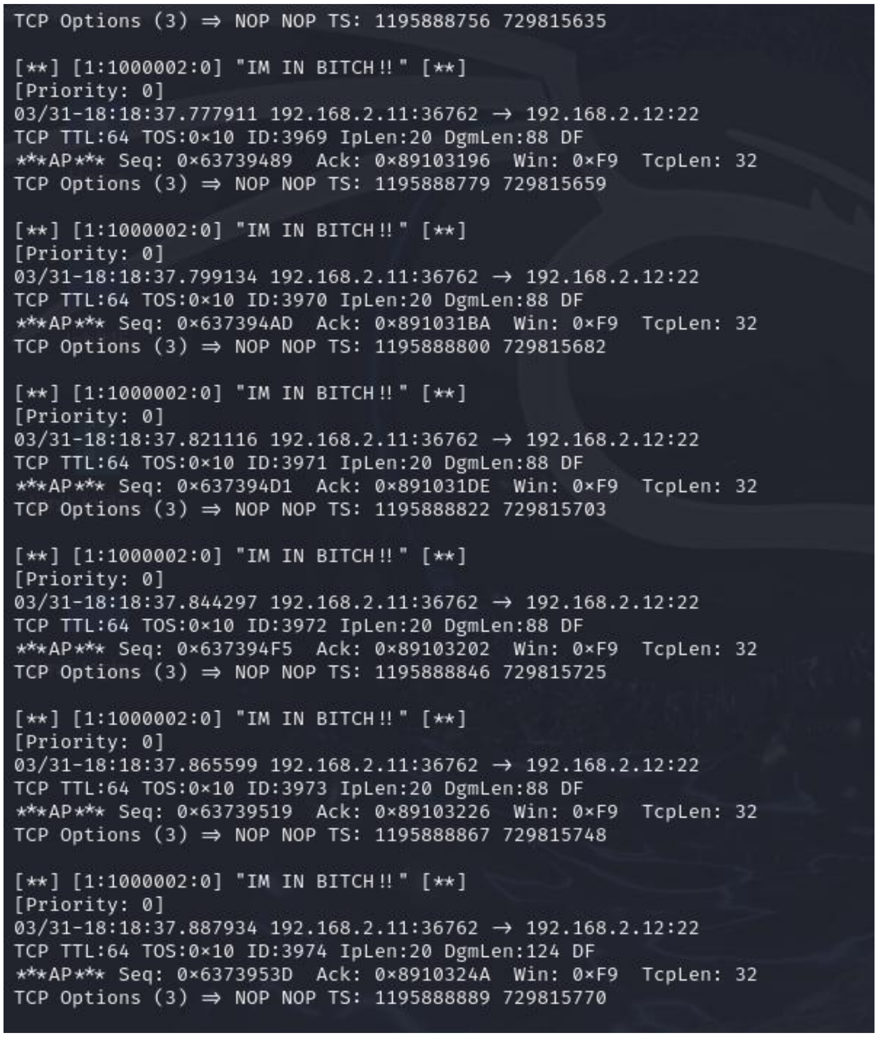


```
 GNU nano 8.2                                             lab10.rules
alert icmp any any -> any any (msg:"PING PONG!!";  sid:1000001;)
alert tcp any any -> any 22 (msg:"IM IN BITCH!!";  sid:1000002;)
```

5) Testez cette r`egle avec snort et montrez qu'elle fonctionne en utilisant la commande ssh. Capture ´ecran de l'alerte snort : (1 point)

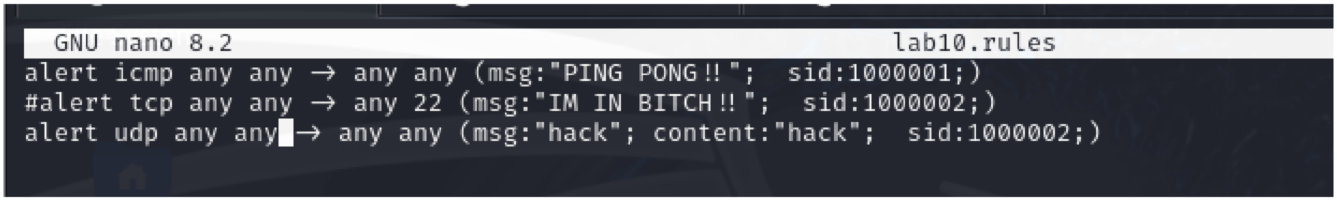


```
┌──(kali㉿kali2024blue)-[~]
└─$ ssh user1@192.168.2.12
user1@192.168.2.12's password:
Welcome to Alpine!

The Alpine Wiki contains a large amount of how-to guides and general
information about administrating Alpine systems.
See <https://wiki.alpinelinux.org/>.

You can setup the system with the command: setup-alpine

You may change this message by editing /etc/motd.

iot:~$ ^C

iot:~$ exit
Connection to 192.168.2.12 closed.

┌──(kali㉿kali2024blue)-[~]
└─$ 
```



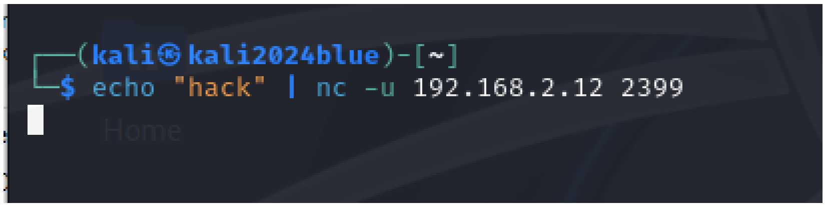
```
TCP Options (3) ⇒ NOP NOP TS: 1195888756 729815635

[**] [1:1000002:0] "IM IN BITCH!!" [**]
[Priority: 0]
03/31-18:18:37.777911 192.168.2.11:36762 → 192.168.2.12:22
TCP TTL:64 TOS:0×10 ID:3969 IpLen:20 DgmLen:88 DF
***AP*** Seq: 0×63739489  Ack: 0×89103196  Win: 0×F9  TcpLen: 32
TCP Options (3) ⇒ NOP NOP TS: 1195888779 729815659

[**] [1:1000002:0] "IM IN BITCH!!" [**]
[Priority: 0]
03/31-18:18:37.799134 192.168.2.11:36762 → 192.168.2.12:22
TCP TTL:64 TOS:0×10 ID:3970 IpLen:20 DgmLen:88 DF
***AP*** Seq: 0×637394AD  Ack: 0×891031BA  Win: 0×F9  TcpLen: 32
TCP Options (3) ⇒ NOP NOP TS: 1195888800 729815682

[**] [1:1000002:0] "IM IN BITCH!!" [**]
[Priority: 0]
03/31-18:18:37.821116 192.168.2.11:36762 → 192.168.2.12:22
TCP TTL:64 TOS:0×10 ID:3971 IpLen:20 DgmLen:88 DF
***AP*** Seq: 0×637394D1  Ack: 0×891031DE  Win: 0×F9  TcpLen: 32
TCP Options (3) ⇒ NOP NOP TS: 1195888822 729815703

[**] [1:1000002:0] "IM IN BITCH!!" [**]
[Priority: 0]
03/31-18:18:37.844297 192.168.2.11:36762 → 192.168.2.12:22
TCP TTL:64 TOS:0×10 ID:3972 IpLen:20 DgmLen:88 DF
***AP*** Seq: 0×637394F5  Ack: 0×89103202  Win: 0×F9  TcpLen: 32
TCP Options (3) ⇒ NOP NOP TS: 1195888846 729815725

[**] [1:1000002:0] "IM IN BITCH!!" [**]
[Priority: 0]
03/31-18:18:37.865599 192.168.2.11:36762 → 192.168.2.12:22
TCP TTL:64 TOS:0×10 ID:3973 IpLen:20 DgmLen:88 DF
***AP*** Seq: 0×63739519  Ack: 0×89103226  Win: 0×F9  TcpLen: 32
TCP Options (3) ⇒ NOP NOP TS: 1195888867 729815748

[**] [1:1000002:0] "IM IN BITCH!!" [**]
[Priority: 0]
03/31-18:18:37.887934 192.168.2.11:36762 → 192.168.2.12:22
TCP TTL:64 TOS:0×10 ID:3974 IpLen:20 DgmLen:124 DF
***AP*** Seq: 0×6373953D  Ack: 0×8910324A  Win: 0×F9  TcpLen: 32
TCP Options (3) ⇒ NOP NOP TS: 1195888889 729815770
```

6) Cr´eez une r`egle snort qui alerte les paquets de type UDP contenant le texte ’hack’. Quelle est cette regle ? (1 point)

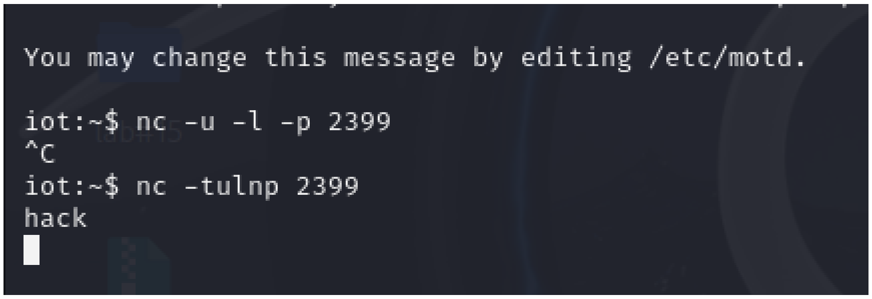


```
  GNU nano 8.2                                   lab10.rules
alert icmp any any → any any (msg:"PING PONG !! ";  sid:1000001;)
#alert tcp any any → any 22 (msg:"IM IN BITCH !! ";  sid:1000002;)
alert udp any any → any any (msg:"hack"; content:"hack";  sid:1000002;)
```

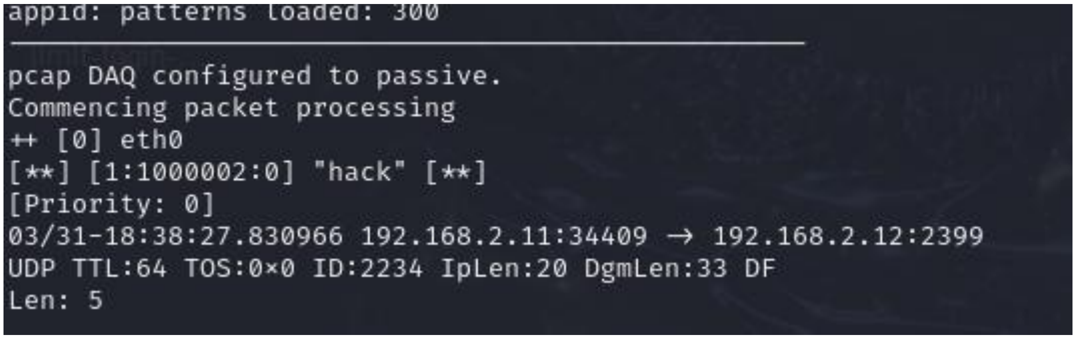
7) Testez cette r`egle avec snort. Montrez qu'elle fonctionne en utilisant un serveur netcat en udp sur la VM snort et un client netcat udp sur Kali. Capture ´ecran de l'alerte snort : (1 point)



```
┌──(kali㉿kali2024blue)-[~]
└─$ echo "hack" | nc -u 192.168.2.12 2399
```

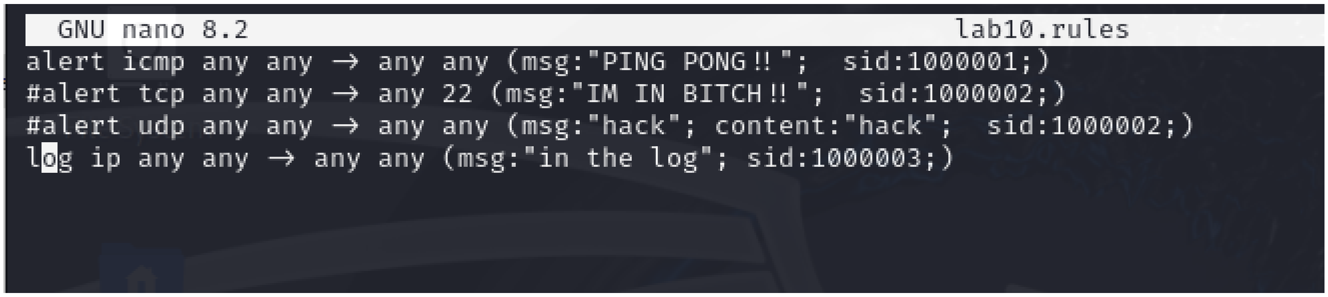


```
You may change this message by editing /etc/motd.

iot:~$ nc -u -l -p 2399
^C
iot:~$ nc -tulnp 2399
hack
```

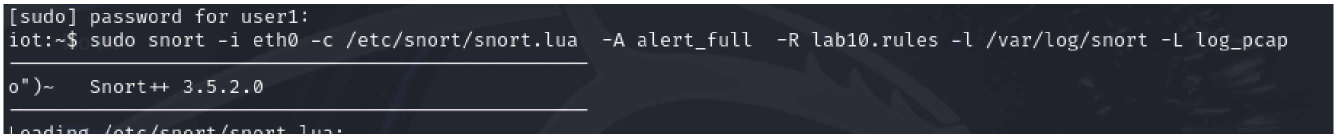


```
appid: patterns loaded: 300
--------------------------------------------------
pcap DAQ configured to passive.
Commencing packet processing
++ [0] eth0
[**] [1:1000002:0] "hack" [**]
[Priority: 0]
03/31-18:38:27.830966 192.168.2.11:34409 → 192.168.2.12:2399
UDP TTL:64 TOS:0×0 ID:2234 IpLen:20 DgmLen:33 DF
Len: 5
```

8) Cr´eez une r`egle snort qui stocke les paquets dans un fichier de type pcap. Quelle est cette regle ? (1 point)

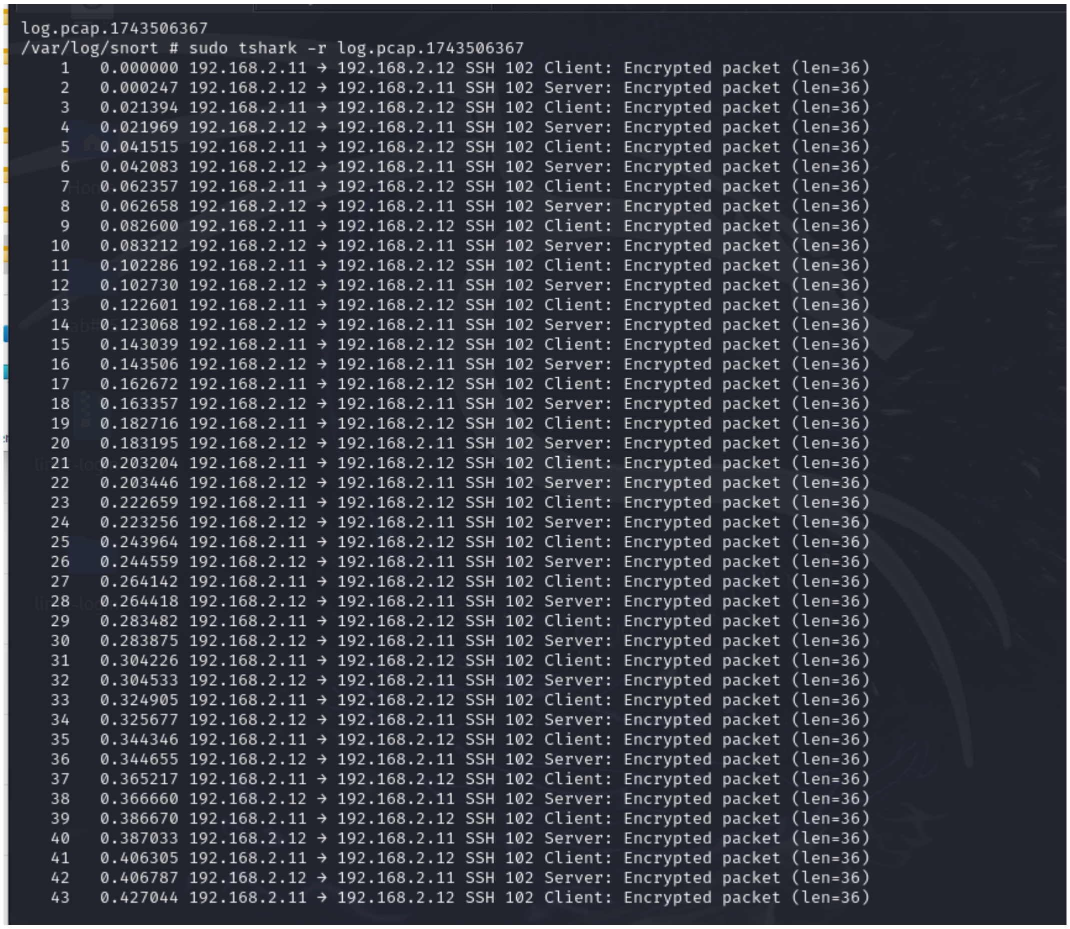


```
  GNU nano 8.2                                   lab10.rules
alert icmp any any → any any (msg:"PING PONG !! ";  sid:1000001;)
#alert tcp any any → any 22 (msg:"IM IN BITCH !! ";  sid:1000002;)
#alert udp any any → any any (msg:"hack"; content:"hack";  sid:1000002;)
log ip any any → any any (msg:"in the log"; sid:1000003;)
```

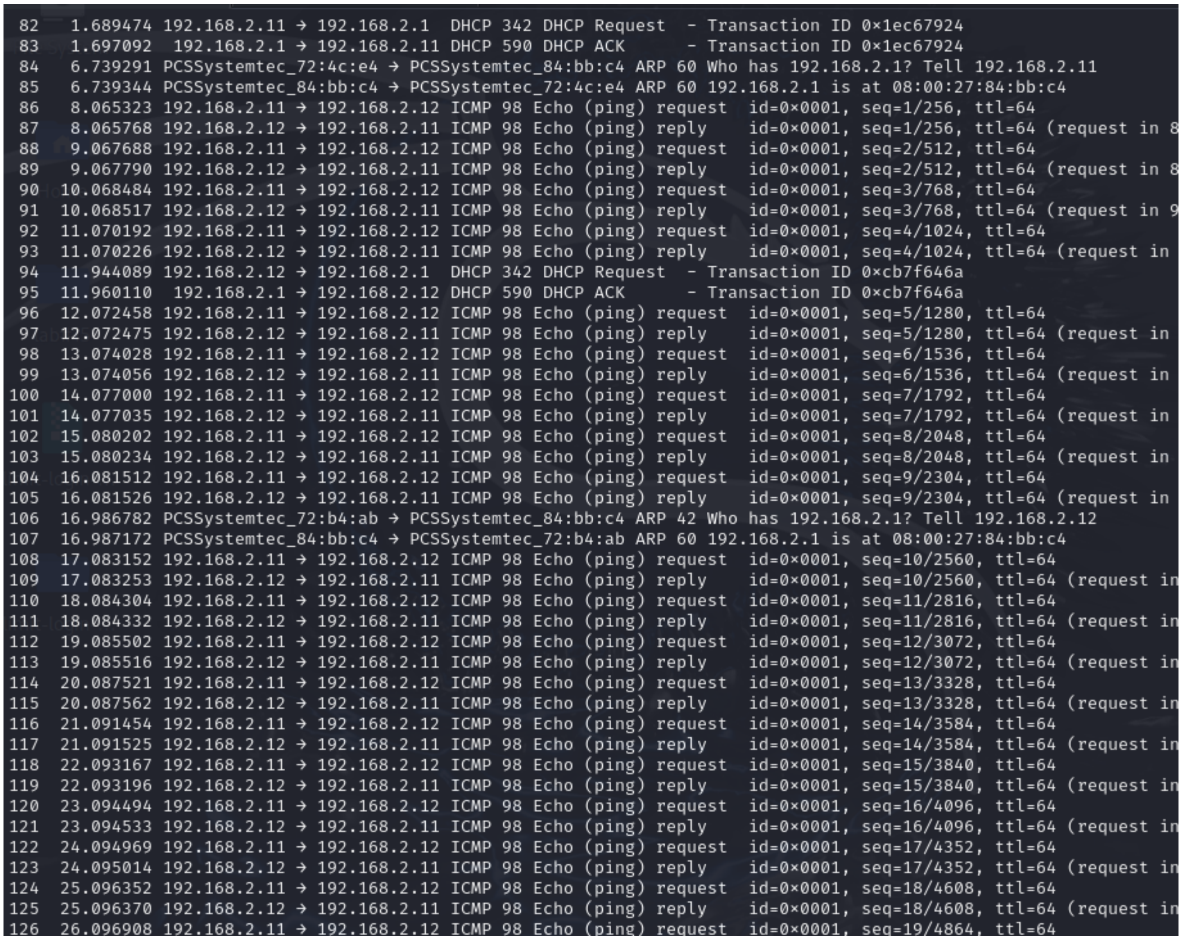


```
[sudo] password for user1:
iot:~$ sudo snort -i eth0 -c /etc/snort/snort.lua  -A alert_full  -R lab10.rules -l /var/log/snort -L log_pcap
--------------------------------------------------
o")~   Snort++ 3.5.2.0
--------------------------------------------------
Loading /etc/snort/snort.lua:
```

9) affichez le contenu de ce fichier (pcap) avec tshark (1 point) Fin du laboratoire



```
log.pcap.1743506367
/var/log/snort # sudo tshark -r log.pcap.1743506367
    1   0.000000 192.168.2.11 → 192.168.2.12 SSH 102 Client: Encrypted packet (len=36)
    2   0.000247 192.168.2.12 → 192.168.2.11 SSH 102 Server: Encrypted packet (len=36)
    3   0.021394 192.168.2.11 → 192.168.2.12 SSH 102 Client: Encrypted packet (len=36)
    4   0.021969 192.168.2.12 → 192.168.2.11 SSH 102 Server: Encrypted packet (len=36)
    5   0.041515 192.168.2.11 → 192.168.2.12 SSH 102 Client: Encrypted packet (len=36)
    6   0.042083 192.168.2.12 → 192.168.2.11 SSH 102 Server: Encrypted packet (len=36)
    7   0.062357 192.168.2.11 → 192.168.2.12 SSH 102 Client: Encrypted packet (len=36)
    8   0.062658 192.168.2.12 → 192.168.2.11 SSH 102 Server: Encrypted packet (len=36)
    9   0.082600 192.168.2.11 → 192.168.2.12 SSH 102 Client: Encrypted packet (len=36)
   10   0.083212 192.168.2.12 → 192.168.2.11 SSH 102 Server: Encrypted packet (len=36)
   11   0.102286 192.168.2.11 → 192.168.2.12 SSH 102 Client: Encrypted packet (len=36)
   12   0.102730 192.168.2.12 → 192.168.2.11 SSH 102 Server: Encrypted packet (len=36)
   13   0.122601 192.168.2.11 → 192.168.2.12 SSH 102 Client: Encrypted packet (len=36)
   14   0.123068 192.168.2.12 → 192.168.2.11 SSH 102 Server: Encrypted packet (len=36)
   15   0.143039 192.168.2.11 → 192.168.2.12 SSH 102 Client: Encrypted packet (len=36)
   16   0.143506 192.168.2.12 → 192.168.2.11 SSH 102 Server: Encrypted packet (len=36)
   17   0.162672 192.168.2.11 → 192.168.2.12 SSH 102 Client: Encrypted packet (len=36)
   18   0.163357 192.168.2.12 → 192.168.2.11 SSH 102 Server: Encrypted packet (len=36)
   19   0.182716 192.168.2.11 → 192.168.2.12 SSH 102 Client: Encrypted packet (len=36)
   20   0.183195 192.168.2.12 → 192.168.2.11 SSH 102 Server: Encrypted packet (len=36)
   21   0.203204 192.168.2.11 → 192.168.2.12 SSH 102 Client: Encrypted packet (len=36)
   22   0.203446 192.168.2.12 → 192.168.2.11 SSH 102 Server: Encrypted packet (len=36)
   23   0.222659 192.168.2.11 → 192.168.2.12 SSH 102 Client: Encrypted packet (len=36)
   24   0.223256 192.168.2.12 → 192.168.2.11 SSH 102 Server: Encrypted packet (len=36)
   25   0.243964 192.168.2.11 → 192.168.2.12 SSH 102 Client: Encrypted packet (len=36)
   26   0.244559 192.168.2.12 → 192.168.2.11 SSH 102 Server: Encrypted packet (len=36)
   27   0.264142 192.168.2.11 → 192.168.2.12 SSH 102 Client: Encrypted packet (len=36)
   28   0.264418 192.168.2.12 → 192.168.2.11 SSH 102 Server: Encrypted packet (len=36)
   29   0.283482 192.168.2.11 → 192.168.2.12 SSH 102 Client: Encrypted packet (len=36)
   30   0.283875 192.168.2.12 → 192.168.2.11 SSH 102 Server: Encrypted packet (len=36)
   31   0.304226 192.168.2.11 → 192.168.2.12 SSH 102 Client: Encrypted packet (len=36)
   32   0.304533 192.168.2.12 → 192.168.2.11 SSH 102 Server: Encrypted packet (len=36)
   33   0.324905 192.168.2.11 → 192.168.2.12 SSH 102 Client: Encrypted packet (len=36)
   34   0.325677 192.168.2.12 → 192.168.2.11 SSH 102 Server: Encrypted packet (len=36)
   35   0.344346 192.168.2.11 → 192.168.2.12 SSH 102 Client: Encrypted packet (len=36)
   36   0.344655 192.168.2.12 → 192.168.2.11 SSH 102 Server: Encrypted packet (len=36)
   37   0.365217 192.168.2.11 → 192.168.2.12 SSH 102 Client: Encrypted packet (len=36)
   38   0.366660 192.168.2.12 → 192.168.2.11 SSH 102 Server: Encrypted packet (len=36)
   39   0.386670 192.168.2.11 → 192.168.2.12 SSH 102 Client: Encrypted packet (len=36)
   40   0.387033 192.168.2.12 → 192.168.2.11 SSH 102 Server: Encrypted packet (len=36)
   41   0.406305 192.168.2.11 → 192.168.2.12 SSH 102 Client: Encrypted packet (len=36)
   42   0.406787 192.168.2.12 → 192.168.2.11 SSH 102 Server: Encrypted packet (len=36)
   43   0.427044 192.168.2.11 → 192.168.2.12 SSH 102 Client: Encrypted packet (len=36)
```



```
 82   1.689474 192.168.2.11 → 192.168.2.1  DHCP 342 DHCP Request  - Transaction ID 0×1ec67924
 83   1.697092  192.168.2.1 → 192.168.2.11 DHCP 590 DHCP ACK      - Transaction ID 0×1ec67924
 84   6.739291 PCSSystemtec_72:4c:e4 → PCSSystemtec_84:bb:c4 ARP 60 Who has 192.168.2.1? Tell 192.168.2.11
 85   6.739344 PCSSystemtec_84:bb:c4 → PCSSystemtec_72:4c:e4 ARP 60 192.168.2.1 is at 08:00:27:84:bb:c4
 86   8.065323 192.168.2.11 → 192.168.2.12 ICMP 98 Echo (ping) request  id=0×0001, seq=1/256, ttl=64
 87   8.065768 192.168.2.12 → 192.168.2.11 ICMP 98 Echo (ping) reply    id=0×0001, seq=1/256, ttl=64 (request in 8
 88   9.067688 192.168.2.11 → 192.168.2.12 ICMP 98 Echo (ping) request  id=0×0001, seq=2/512, ttl=64
 89   9.067790 192.168.2.12 → 192.168.2.11 ICMP 98 Echo (ping) reply    id=0×0001, seq=2/512, ttl=64 (request in 8
 90  10.068484 192.168.2.11 → 192.168.2.12 ICMP 98 Echo (ping) request  id=0×0001, seq=3/768, ttl=64
 91  10.068517 192.168.2.12 → 192.168.2.11 ICMP 98 Echo (ping) reply    id=0×0001, seq=3/768, ttl=64 (request in 9
 92  11.070192 192.168.2.11 → 192.168.2.12 ICMP 98 Echo (ping) request  id=0×0001, seq=4/1024, ttl=64
 93  11.070226 192.168.2.12 → 192.168.2.11 ICMP 98 Echo (ping) reply    id=0×0001, seq=4/1024, ttl=64 (request in
 94  11.944089 192.168.2.12 → 192.168.2.1  DHCP 342 DHCP Request  - Transaction ID 0×cb7f646a
 95  11.960110  192.168.2.1 → 192.168.2.12 DHCP 590 DHCP ACK      - Transaction ID 0×cb7f646a
 96  12.072458 192.168.2.11 → 192.168.2.12 ICMP 98 Echo (ping) request  id=0×0001, seq=5/1280, ttl=64
 97  12.072475 192.168.2.12 → 192.168.2.11 ICMP 98 Echo (ping) reply    id=0×0001, seq=5/1280, ttl=64 (request in
 98  13.074028 192.168.2.11 → 192.168.2.12 ICMP 98 Echo (ping) request  id=0×0001, seq=6/1536, ttl=64
 99  13.074056 192.168.2.12 → 192.168.2.11 ICMP 98 Echo (ping) reply    id=0×0001, seq=6/1536, ttl=64 (request in
100  14.077000 192.168.2.11 → 192.168.2.12 ICMP 98 Echo (ping) request  id=0×0001, seq=7/1792, ttl=64
101  14.077035 192.168.2.12 → 192.168.2.11 ICMP 98 Echo (ping) reply    id=0×0001, seq=7/1792, ttl=64 (request in
102  15.080202 192.168.2.11 → 192.168.2.12 ICMP 98 Echo (ping) request  id=0×0001, seq=8/2048, ttl=64
103  15.080234 192.168.2.12 → 192.168.2.11 ICMP 98 Echo (ping) reply    id=0×0001, seq=8/2048, ttl=64 (request in
104  16.081512 192.168.2.11 → 192.168.2.12 ICMP 98 Echo (ping) request  id=0×0001, seq=9/2304, ttl=64
105  16.081526 192.168.2.12 → 192.168.2.11 ICMP 98 Echo (ping) reply    id=0×0001, seq=9/2304, ttl=64 (request in
106  16.986782 PCSSystemtec_72:b4:ab → PCSSystemtec_84:bb:c4 ARP 42 Who has 192.168.2.1? Tell 192.168.2.12
107  16.987172 PCSSystemtec_84:bb:c4 → PCSSystemtec_72:b4:ab ARP 60 192.168.2.1 is at 08:00:27:84:bb:c4
108  17.083152 192.168.2.11 → 192.168.2.12 ICMP 98 Echo (ping) request  id=0×0001, seq=10/2560, ttl=64
109  17.083253 192.168.2.12 → 192.168.2.11 ICMP 98 Echo (ping) reply    id=0×0001, seq=10/2560, ttl=64 (request in
110  18.084304 192.168.2.11 → 192.168.2.12 ICMP 98 Echo (ping) request  id=0×0001, seq=11/2816, ttl=64
111  18.084332 192.168.2.12 → 192.168.2.11 ICMP 98 Echo (ping) reply    id=0×0001, seq=11/2816, ttl=64 (request in
112  19.085502 192.168.2.11 → 192.168.2.12 ICMP 98 Echo (ping) request  id=0×0001, seq=12/3072, ttl=64
113  19.085516 192.168.2.12 → 192.168.2.11 ICMP 98 Echo (ping) reply    id=0×0001, seq=12/3072, ttl=64 (request in
114  20.087521 192.168.2.11 → 192.168.2.12 ICMP 98 Echo (ping) request  id=0×0001, seq=13/3328, ttl=64
115  20.087562 192.168.2.12 → 192.168.2.11 ICMP 98 Echo (ping) reply    id=0×0001, seq=13/3328, ttl=64 (request in
116  21.091454 192.168.2.11 → 192.168.2.12 ICMP 98 Echo (ping) request  id=0×0001, seq=14/3584, ttl=64
117  21.091525 192.168.2.12 → 192.168.2.11 ICMP 98 Echo (ping) reply    id=0×0001, seq=14/3584, ttl=64 (request in
118  22.093167 192.168.2.11 → 192.168.2.12 ICMP 98 Echo (ping) request  id=0×0001, seq=15/3840, ttl=64
119  22.093196 192.168.2.12 → 192.168.2.11 ICMP 98 Echo (ping) reply    id=0×0001, seq=15/3840, ttl=64 (request in
120  23.094494 192.168.2.11 → 192.168.2.12 ICMP 98 Echo (ping) request  id=0×0001, seq=16/4096, ttl=64
121  23.094533 192.168.2.12 → 192.168.2.11 ICMP 98 Echo (ping) reply    id=0×0001, seq=16/4096, ttl=64 (request in
122  24.094969 192.168.2.11 → 192.168.2.12 ICMP 98 Echo (ping) request  id=0×0001, seq=17/4352, ttl=64
123  24.095014 192.168.2.12 → 192.168.2.11 ICMP 98 Echo (ping) reply    id=0×0001, seq=17/4352, ttl=64 (request in
124  25.096352 192.168.2.11 → 192.168.2.12 ICMP 98 Echo (ping) request  id=0×0001, seq=18/4608, ttl=64
125  25.096370 192.168.2.12 → 192.168.2.11 ICMP 98 Echo (ping) reply    id=0×0001, seq=18/4608, ttl=64 (request in
126  26.096908 192.168.2.11 → 192.168.2.12 ICMP 98 Echo (ping) request  id=0×0001, seq=19/4864, ttl=64
```